wIDS: a multilayer IDS for Wireless-based SCADA Systems

Lyes Bayou, David Espes, Nora Cuppens-Boulahia, Frédéric Cuppens

wIDS: a multilayer IDS for Wireless-based SCADA Systems

Lyes Bayou[1], David Espes[2], Nora Cuppens-Boulahia[1], and Frédéric Cuppens[1]

[1] IMT Atlantique-LabSTICC, 2 Rue de la Châtaigneraie, Césson Sévigné, France
[2] University of Western Brittany - LabSTICC, Brest, France

Abstract. The increasing use of wireless sensors networks in Supervisory Control and Data Acquisition systems (SCADA) raises the need of enforcing the security of this promising technology. Indeed, SCADA systems are used to manage critical installations that have hard security, reliability and real-time requirements. Consequently, in order to ensure Wireless Industrial Sensor Networks (WISN) security, Intrusion Detection Systems should be used as a second line of defense, in addition to sensor's embedded security mechanisms. In this paper, we present wIDS a multilayer specification-based Intrusion Detection System specially tailored for WISN. It has a two-level detection architecture and is based on a formal description of node's normal behavior.

1 Introduction

Wireless Industrial Sensor Network (WISN) are now established as a widely used technology in industrial environments. Indeed, comparing to wired technologies, they allow significant decreases in deployment and maintenance costs. In the same time, they increase the system sensing capabilities as wireless sensors can be deployed in hardly reachable and adversarial environments.

In industrial environments, Supervisory Control and Data Acquisition systems (SCADA) are used for monitoring and managing complex installations such as power plants, refineries, railways, etc. These systems rely on sensors deployed over large area to gather in real time information about the industrial process. These informations are sent to a controller that processes them and sent back commands to field devices such as actuators or valves.

Security is an important issue in SCADA systems. Indeed, the disruption of these systems can cause significant damages to critical infrastructures such as electric power distribution, oil and natural gas distribution, water and wastewater treatment, and transportation systems. This can have a serious impact on public health, safety and can lead to large economical losses.

On the other hand, ensuring security in WISN is a challenging task. Indeed, WISN are subject to the same attacks as other wireless networks. Mainly, attackers use wireless communication as a vector to launch their attacks. Furthermore,

sensor's limited capabilities in terms of processing power, memory space and energy make hard the implementation of strong security mechanisms.

Thus, in addition to sensor's embedded mechanisms that ensure authentication, confidentiality and availability, Intrusion Detection Systems can be used as a second line of defense for enforcing the overall system security and in particular for detecting unknown attacks.

The exchanged traffic in a SCADA system is highly predictable in terms of amount and frequency. Indeed, it involves limited human interaction and is mainly composed of automated devices that execute defined actions at defined times. Therefore, by modeling the normal expected behavior of wireless nodes, we can detect malicious action as being actions deviating from the established model.

In this paper, we present wIDS, a multilayer specification-based Intrusion Detection System specially tailored for Wireless Industrial Sensor Networks. The proposed IDS checks the compliance of each action performed by a wireless node towards the formal model of the expected normal behavior. To do that, access control rules are used for modeling authorized actions that a wireless node can perform. These rules are mainly built on the base of the specifications of each layer of the communication protocol, node's localization and the industrial process configuration. They also take into consideration the capabilities and limitations of the wireless nodes . Thus, by specifying security policy at an abstract level, we are able to define and manage more accurate and efficient security rules independently from nodes and network characteristics such as sensor natures and density or the network topology. Then, these characteristics are used later when deriving concrete security rules. Also, in addition to alerts that are raised by actions deviating from the normal model, we define additional intrusion rules that aim to detect basic attacker actions such as injecting, deleting, modifying and delaying packets.

The rest of the paper is organized as follows. Section 2 presents previous works done in this field and emphasis their limits. In Section 3, we present briefly the protocol WirelessHART that is the use case of this study. We present in Section 4, the proposed IDS for Wireless Industrial Sensor Networks. We describe its two-level detection architecture and present the formalism used to build node's normal behavior. Section 5 details security rules defined on the base of WirelessHART specifications. In Section 6, we present how detection rules are defined to detect suspicious actions. The performances of the proposed wIDS are presented and discussed in Section 7. Finally, Section 8 presents the conclusion and future works.

2 Related Work

In the literature, there are only few studies on the security of Wireless Sensor Networks used in industrial environments. Mainly, proposed solutions for SCADA systems focus on applying IDS techniques to wired-based networks [1–3] and neglect those using wireless communications.

More generally, several IDS are proposed for generic WSN [4]. However, proposed solutions are not suitable for WISN. Firstly, WISN have more hard requirements than generic WSN such as real-time and reliable communications. Indeed, dropped or delayed data may lead to physical losses. Secondly, these proposed IDS are mainly restricted to detect specific kinds of attack while WISN must be secured towards a broad spectrum of known and unknown attacks. Nevertheless, these works should be considered in order to propose a solution designed for WISN.

Thus, in [5], Da Sila et al. propose one of the first intrusion detection systems for WSN. They designed a decentralized system in which a set of nodes is designated as *monitor* and is responsible of monitoring their neighbors. The proposed IDS is based on the statistical inference of the network behavior. It only monitors data messages and ignores other kinds of exchanged messages. It includes seven types of rules that aim to detect common attacks.

For its part, Roosta et al. propose in [6] an intrusion detection system for wireless process control systems. The system consists of two components: a central IDS and multiple field IDS that passively monitor communications in their neighborhood. They periodically send collected data to the central IDS that checks their conformity with the security policy. This IDS models normal behavior of the different wireless devices on the base of some network specifications and traffic characteristics inferred statistically. Attacks are detected when there is a deviation from the model. However, it defines a few numbers of rules (8 rules) that do not cover all well-known attacks. Furthermore, as the detection logic is centralized, this solution requires continuous communications with field IDS which can add a significant network overload.

In [7], Coppolino et al. propose an architecture for an intrusion detection system for critical information infrastructures using wireless sensor network. Their solution is a hybrid approach combining misuse and anomaly based techniques. It is composed of a Central Agent and several IDS Local Agents that monitors exchanged messages in their neighborhood. They calculate a statistical model of exchanged traffic and raise a temporary alert when nodes actions deviate from this model. The central agent combines these alerts and confirms them on the base of misuse rules. This IDS focuses on attacks against routing protocols and detects only two kinds of attacks i.e., sinkhole and sleep deprivation attacks.

Shin et al. [8] propose a hierarchical framework for intrusion detection for WISN. It is based on two-level clustering; multihop clusters for data aggregation and one-hop clusters for intrusion detection. This results in a four layer hierarchy: member nodes (MN) are the leaves, cluster heads (CH) manage MNs, gateways (GW) bundle clusters and a base station (BS) is the root of the hierarchy. These different levels implement the same detection logic, however they respond differently. Thus, MN only report to CH while other roles have the ability to react to attacks.

In our study, we aim to propose a solution that is able to detect either known and unknown attacks. Furthermore, such solution should have a multilevel detection architecture to monitor both local and end-to-end communications (gener-

ally encrypted) and also in order to provide global coordination. Low detection level should have full detection capabilities in order to avoid overloading the network by additional exchanges and to have quick and accurate detections.

3 Backgrounds on WirelessHART

In our work, we choose to apply our approach to WISN that implements WirelessHART and we will use it as a common thread all along this paper. Indeed, WirelessHART [9] is the first standardized wireless communication protocol specially developed for industrial process management. It is included in version 7 of the HART standard, released in 2007, and was approved as an international standard in 2010 (IEC 62591).

A WirelessHART network, illustrated in Figure 1, has the following characteristics:

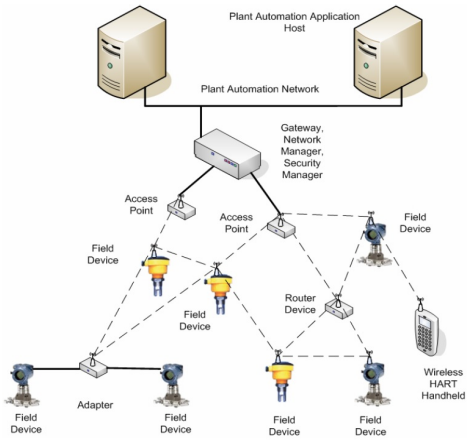

Fig. 1. Example of a WirelessHART network [10]

- A Gateway that connects the wireless network to the plant automation network, allowing data to flow between the two networks. It can also be used to convert data and commands from one protocol to another one;
- A Network Manager that is responsible for the overall management, scheduling, and optimization of the wireless network. It generates and maintains all of the routing information and also allocates communication resources;
- A Security Manager that is responsible for the generation, storage, and management of cryptographic keys;
- Access Points that connect the Gateway to the wireless network through a wired connection;

- Field devices deployed in the plant field and which can be sensors or actuators;
- Routers used for forwarding packets from one network device to another;
- Handheld devices that are portable equipments operated by the plant personnel used in the installation and during the maintenance of network devices.

WirelessHART implements several mechanisms to ensure security in both hop-by-hop and end-to-end communication. In hop-by-hop transmission, security is provided by the Data Link Layer (DLL) using a cryptographic key called "Network Key" shared by all devices composing the wireless network. Each packet is authenticated using a keyed Message Integrity Code (MIC).

The end-to-end security is provided by the Network Layer (NL) using a cryptographic key called "Session Key" known only by the two communicant devices. Packets are both authenticated by a MIC and their payload is also encrypted.

4 Multilayer specification-based IDS

In this Section, we present *wIDS* a multilayer specification based IDS for securing Wireless Industrial Sensors Networks. We describe its architecture, its components and its analyzing process.

Specification-based intrusion detection approaches formally define the model of legitimate behavior and raise intrusion alerts when user's actions deviate from the model [3] [4]. WISN are composed of nodes that have a predictable behavior and involves few human interactions. Consequently, on the base of the communication protocol specifications, the process configuration and wireless nodes capabilities, we can build an accurate model of the expected nodes's behavior.

We should also note that specification-based intrusion detection system do not require any training step. Therefore, they can be applied and used directly.

In this study, we assume that the aim of the attacker is to disturb the industrial process. This goal can be achieved by dropping some packets, injecting into the network false packets or modifying packets during their transmission. Furthermore, the attacker can also choose to delay the transmission of some important packets (alarms, sensing data, etc) in order to lead the process to an uncertain state or to hide its malicious actions. Therefore, we consider an attacker that can intercept, modify, forge or delay packets. It can be an insider or an outsider attacker.

4.1 wirelessOrBAC

We propose in this work wirelessOrBAC, an OrBAC [11] extension that we develop in order to efficiently model Wireless Sensor Networks specifications. OrBAC has already been used to specify network security policy, especially in firewall management [12], intrusion detection (IDS) and intrusion prevention systems (IPS) [13]. It allows the definition of a conflict-free security policy and

by using the concept of *context*, it makes it possible to define dynamic rules that fit the system's changes. This extension allows using access control rules, modeling authorized actions and the expected behavior of wireless nodes. Then, using this model, we can detect malicious actions by checking the compliance of each node action toward the defined security policy.

Furthermore, wirelessOrBAC allows the definition of both access control rules and intrusion detection rules using the same formalism.

To define a security policy, wirelessOrBAC defines the following concepts:

- wNetwork: It is an abstraction of the considered WSN.
- wRole: It is an abstraction of predefined roles used in WSN such as: wSensor, wSink, wForwarder, wCluster_head.
- wDevice: It is an abstraction of wireless devices. It is composed of one or several wRole.
- wActivity: It is an abstraction of wireless actions that a wDevice can perform such as: sending, receiving, forwarding and aggregating.
- wView: It is an abstraction of messages sent from a wDevice to another one.
- wContext: It is used to model extra conditions that a subject, an action and an object must satisfy to activate a security rule.

Thus, a security rule is defined as follows:

$$wRule(security\_rule, wnet, d, a, v, c) \tag{1}$$

that means that in wNetwork *wnet,* wDevice d is granted *security_ rule* to perform wActivity a on wView v within wContext c and where *security_ rule* belongs to {perm, prohib, obl} (Corresponding to: permission, prohibition and obligation).

Finally, concrete security rules are derived as follows:

$$\begin{aligned} &wRule(perm, wnet, wRole, wActivity, wView, c) \\ &\wedge empower(wnet, s, wRole) \wedge consider(wnet, a, wActivity) \\ &\wedge use(wnet, o, wView) \wedge hold(wnet, s, a, o, c) \\ &\rightarrow Is\_Permitted(s, a, o) \end{aligned} \tag{2}$$

where:

- *empower(wnet, s, wRole)*: means that in wNetwork *wnet*, subject s is empowered in *wRole*.
- *consider(wnet, a, wActivity)*: means that in wNetwork *wnet*, action a is considered an implementation of *wActivity*.
- *use(wnet, o, wView)*: means that in wNetwork *wnet*, object o is used in *wView*.
- *hold(wnet, s, a, o, c)*: means that in wNetwork *wnet*, *wContext* c is active for s, a and o.

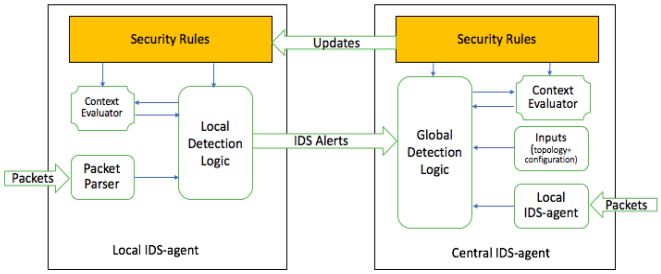

Fig. 2. The Central-IDS and IDS-Agents Architecture

4.2 wIDS architecture

As indicated in Fig. 2, wIDS has a two-level architecture consisting in a central-IDS agent and several IDS-agents.

- The central-IDS agent: It is implemented in the Network Manager (resp. in the sink) in the case of wirelessHART (resp. in the case of a WISN). In addition of playing the role of an IDS-agent in its neighborhood, it monitors end-to-end communications after they are decrypted. It may check routing tables and transmission scheduling consistency; and performs global coordination between IDS-agents.
- IDS-agents: They are implemented in selected sensor nodes. They are responsible for monitoring local communications of sensor nodes inside their neighborhood. They listen in promiscuous mode to all packets exchanged in their neighborhood. Then, they extract from them, relevant informations in order to check their compliance with the security policy.

The abstract security policy is defined at the central-IDS agent using the wirelessOrBAC formalism. It is also provided with several inputs such as node localizations, industrial process parameters and nodes configuration. The central-IDS agent provisions IDS-agents with security rules and several inputs related to nodes available in their neighborhood (i.e., the list of monitored nodes). It also updates if necessary all these information. Each IDS-agent is in charge of the application of the security policy in its area and alerts the central-IDS agent when policy violation occurs.

4.3 IDS-agents deployment scheme

The scheme used for the deployment of IDS-agents, is an important issue. Indeed, as WSN are decentralized systems, the localization of monitoring devices must be chosen carefully otherwise a part of exchanged traffic will not be monitored.

The deployment of IDS-agents is out of the scope of this paper. However, in our study, we use the scheme proposed in [14]. This scheme uses the graph theory

concept of *Connected Dominated Set* to ensure the gathering of the whole exchanged traffic. Also, it presents the following characteristics: *a)* each IDS-agent is able to detect basic attacks occurring in its neighborhood without any cooperation with other IDS-agents; *b)* it creates a secure and reliable communication channel between each IDS-agent and the central-IDS; *c)* it requires an acceptable IDS-agents number to ensure an efficient network monitoring and coverage.

In wIDS, IDS-agents are implemented in sensors with enhanced capabilities. These nodes, called *Super-Nodes*, will act as classical sensor nodes by fulfilling sensing tasks and implements in the same time the detection logic. By using the aforementioned deployment scheme, selected *Super-Nodes* represents between 20%-25% of the total network nodes number.

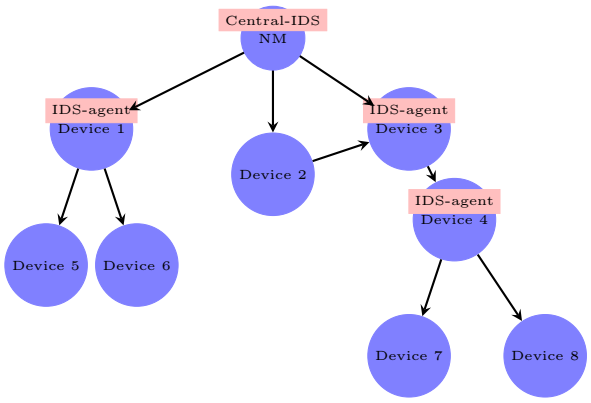

Fig. 3. IDS-agents deployment

5 Expected behavior modeling rules

In this Section, based on WirelessHART specification [9], we define using wirelessOrBAC rules the expected node's normal behavior. These rules express authorized actions at each protocol layer. We gather them in several categories and present hereafter, examples of each of them.

5.1 Meshed wireless network rules

In a wirelessHART network, all devices have the capability to forward packets of devices that are located several hops away from the Network Manager. That means that a device can send packets to any of its neighbors:

$$wRule(perm, wnet, wDevice, sending, packets, neighbors) \quad (3)$$

where *neighbors* is a wContext indicating that $s1$ performs action *sending* object *packet* to node $s2$ that is in its neighborhood:

$$\begin{aligned} Hold(WSN, s1, sending, packet, neighbors) \\ \leftarrow is\_dstAddr(packet, s2) \wedge is\_neighbor(s1, s2) \end{aligned} \tag{4}$$

5.2 Packets construction rules

The WirelessHART specifications give guidelines on how packets should be built and the possible values of each field. Thus, the length and value of each field must be checked. Also, fields of the same packet must be consistent with each other.

$$\begin{aligned} wRule(perm, wnet, wDevice, sending, packet, \_) \\ \leftarrow is\_ValidPacket(packet) \end{aligned} \tag{5}$$

where $is\_ValidPacket(packet)$ is a predicate indicating if *packet* fulfills wirelessHART construction rules. The symbol "_" indicates that this rule is valid for any wContext.

5.3 Communication level

WirelessHART defines 5 packet types: *Ack*, *Advertise*, *Keep-Alive*, *Disconnect* and *Data* packets. The first 4 types are generated and processed in the Data Link Layer and are not propagated to the Network Layer or forwarded through the network. This means that these packets are only used in local communication between neighbors.

$$\begin{aligned} wRule(perm, wnet, wDevice, sending, packet, neighbors) \\ \leftarrow is\_packetType(packet, Ack|Advertise|keepAlive|Disconnect|Data) \end{aligned} \tag{6}$$

The Data packet type is the only kind of packet that is transmitted in an end-to-end communication. This means that only data packets can be *forwarded* throughout the network:

$$\begin{aligned} wRule(perm, wnet, wDevice, forwarding, packet, \_) \\ \leftarrow is\_packetType(packet, Data) \end{aligned} \tag{7}$$

On the other hand, data packets are only exchanged between the Network Manager ($wSink$) and wireless sensors ($wSensor$) in both sens. This means that a data packet is never send from a wireless sensor to another wireless sensor. Thus:

$$\begin{aligned} wRule(perm, wnet, wSensor, sending, packet, \_) \\ \leftarrow is\_packetType(packet, Data) \\ \wedge is\_dstNLAddr(packet, s2.addr) \wedge empower(wnet, s2, wSink) \end{aligned} \tag{8}$$

and

$$\begin{aligned} wRule(perm, wnet, wSink, sending, packet, \_) \\ \leftarrow is\_packetType(packet, Data) \\ \wedge is\_dstNLAddr(packet, s2.addr) \wedge empower(wnet, s2, wSensor) \end{aligned} \tag{9}$$

5.4 Communication scheduling rules

WirelessHART uses *Time Division Multiple Access* (TDMA) and *Channel hopping* to control the access to the wireless medium. The time is divided in consecutive periods of the same duration called *slots*. Each communication between two devices occurs in one slot of 10 ms.

Typically, two devices are assigned to one time slot (one as the sender and a second as the receiver). Only one packet is transmitted in one slot from the sender to the receiver which has to reply with an acknowledgment packet in the same slot.

In addition, WirelessHART uses channel hopping to provide frequency diversity and avoid interferences. Thus, the 2.4 GHz band is divided into 16 channels numbered from 11 to 26 which provide up to 15 communications in the same slot (Channel 26 is not used). Thus, we have the following rules:

$$wRule(perm, wnet, wDevice, sending, packets, startSlot \wedge assignedFq) \quad (10)$$

where *startSlot* is a wContext indicating that s performs action *sending* object *packet* when a slot time assigned to s starts:

$$\begin{aligned} Hold(WSN, s, sending, packet, startSlot) \\ \leftarrow is\_slotStartTime(s, packet) \end{aligned} \quad (11)$$

and *assignedFq* is a wContext indicating that s uses its assigned frequency when performing action *sending* object *packet* :

$$\begin{aligned} Hold(WSN, s, sending, packet, assignedFq) \\ \leftarrow is\_assignedFq(s, packet) \end{aligned} \quad (12)$$

For the acknowledgment, we have the following rule:

$$wRule(perm, wnet, wDevice, sending, packet, sendingAck) \quad (13)$$

where *sendingAck* is a wContext indicating that $s1$ performs action *sending* object $packet'$ when $s1$ received *packet* from s (at time t), and $packet'$ is destined to s and of type *ack* and $s1$ uses the slot and frequency assigned to s:

$$\begin{aligned} Hold(WSN, s1, sending, packet', sendingAck) \\ \leftarrow packet\_received(s1, packet, t) \wedge is\_srcAddr(packet, s) \\ \wedge is\_packetType(packet', Ack) \wedge is\_dstAddr(packet', s) \\ \wedge is\_assignedFq(s, packet) \wedge is\_slotStartTime(s, packet) \end{aligned} \quad (14)$$

We should note that the Network Manager is responsible of building, managing and updating slots and frequencies planning.

5.5 Packets transmission rules

Sensor nodes are configured to send different kind of packets (i.e., sensing data, keep-alive, advertisement) at a defined time. Thus, sensing data must be sent periodically to the Network Manager.

$$wRule(perm, wnet, wDevice, sending, packet, packetPeriodicity) \leftarrow is\_packetType(packet, Data) \quad (15)$$

where *packetPeriodicity* is a temporal context indicating that s performs action *sending* object *packet* in the planned sending time:

$$Hold(WSN, s, sending, packet, packetPeriodicity) \leftarrow is\_packetPeriodicity(s, packet) \quad (16)$$

5.6 Packets forwarding rules

A wirelessHART network has a meshed topology. Thus, wireless devices that are located several hops from the Network Manager, relay on their neighbors for forwarding their packets to their final destination.

$$wRule(perm, wnet, wDevice, forwarding, packet, ForwardPacket) \quad (17)$$

where *ForwardPacket* is a provisional context (i.e., based on previous device actions) indicating that subject s received object *packet* (at time t) and s must forward this object:

$$Hold(WSN, s, forwarding, packet, ForwardPacket) \leftarrow packet\_received(s, packet, t) \wedge is\_toBeForwarded(packet) \wedge empower(org, s, forwarder) \quad (18)$$

5.7 Routing rules

WirelessHART uses graphs as routing method. A graph consists in a set of directed paths that connect network devices. It is build by the Network manager based on its knowledge of the network topology and connectivity. Every graph has a unique graph identifier that is inserted in the network packet header. Each device receiving this packet, forwards it to the next hop belonging to that graph.

$$wRule(perm, wnet, wDevice, sending, packets, graphNextHop) \quad (19)$$

where *graphNextHop* is a spatial context indicating that s performs *sending* object *packet* to $s2$ that is the next hop of s following graph g:

$$Hold(WSN, s, sending, packet, graphNextHop) \leftarrow is\_dstAddr(packet, s2.addr) \wedge is\_usedGraph(packet, g) \wedge is\_NextHop(s.addr, g, s2.addr) \quad (20)$$

In graph routing there are two kinds of graphs: an *upstream* graph directed from all devices to the Network Manager and several *downstream* graphs directed

from the Network Manager to each device. Thus, sensor nodes use the *upstream* graph for sending packets to the Network Manager:

$$\begin{aligned} wRule(perm, wnet, wSensor, sending, packet, \_) \\ \leftarrow is\_packetType(packet, Data) \\ \wedge is\_usedGraph(packet, g) \wedge is\_upStream(g) \end{aligned} \tag{21}$$

and the Network Manager uses *downstream* graphs for sending packets to sensors:

$$\begin{aligned} wRule(perm, wnet, wSink, sending, packet, \_) \\ \leftarrow is\_packetType(packet, Data) \\ \wedge is\_usedGraph(packet, g) \wedge is\_downStream(g) \end{aligned} \tag{22}$$

5.8 Cross layer consistency rules

As indicated in Section 5.2, packet's fields must complain with the protocol specifications and also be consistent between them. This verification is done according to each layer rules. However, an attacker can bypass this verification by giving contradictory information that fulfills each layer rules. Therefore, for some fields a cross layer verification must be applied. For example, in the case of routing information, DLL and NL fields must be consistent:

$$\begin{aligned} wRule(perm, wnet, wSensor, sending, packet, \_) \\ \leftarrow is\_dstAddr(packet, s1.addr) \wedge is\_dstNLAddr(packet, s2.addr) \\ \wedge is\_usedGraph(packet, g) \wedge is\_NextHop(s1.addr, g, s2.addr) \end{aligned} \tag{23}$$

6 wIDS detection rules

In order to detect malicious actions, wIDS apply a close policy requirement. This means that each action initiated by wireless nodes is compared to the defined security policy and a security alert is raised if the verification failed. Thus, all node's actions must:

1. be explicitly allowed by the security policy;
2. and is compliant with all rules that match the action.

Thus each IDS-agent implements Algo. 1 in order to check the compliance of actions performed by wireless nodes.

Thus, each time a node s performs an action a on object o, we first build M the set of security rules that matches the tuple {s,a,o}. If the set M is empty this indicates that there is not a security rule that explicitly permits that s performs an action a on object o. Otherwise, the tuple {s,a,o} is compared towards each rule $m \in M$ to check if it is compliant with that rule (see Rule 2). If the tuple {s,a,o} is not compliant with a security rule m, it is considered as a malicious action and an intrusion rule is raised. Else, if the tuple {s,a,o} is compliant with all security rules $m \in M$, it is considered as a legitimate action.

Algorithm 1 Conformity checking algorithm

```
1: procedure ACTIONVALIDATION(s, a, o)          ▷ subject s performs action a on o
2:     M=matchingRule(s,a,o);                   ▷ Build M the set of rules matching s,a and o
3:     if M is empty then
4:         return false;                        ▷ s is not permitted to perform action a on o
5:     end if
6:     validAction ← true;
7:     while M is not empty ∧ validAction do    ▷ repeat until all rules are checked
8:         Select m from the set M;
9:         M = M − {m};
10:        validAction ← checkValidity(s,a,o,m);   ▷ checks if rule m allows that s
   performs a on o
11:    end while
12:    return validAction
13: end procedure
```

6.1 Default IDS alert

We chose to model IDS alert as *wContexts*. This permits not only the accurate identification of the malicious action but also can allow an automatic reaction by for example the activation or deactivation of some security rules. It also allows the global coordination of alerts in the central-IDS.

To do that, we define *idsAlertCtx* a default context that is activated when an action performed by a wireless node violates a security rule defined in Section 5:

$$\begin{aligned} &\forall s \in S, o \in O, a \in A, \\ &Hold(wnet, s, a, o, idsAlertCtx) \leftarrow \\ &Action(s, a, o) \wedge \neg actionValidation(s, a, o) \end{aligned} \tag{24}$$

where *Action(s,a,o)* indicates that subject s performed action a on object o and $\neg actionValidation(s, a, o)$ (See Algo. 1) indicates that $Action(s, a, o)$ do not match any defined security rules.

6.2 Basic malicious action IDS alert

In order to enforce wIDS detection capabilities, we define additional security rules that aim to detect basic actions that an attacker can perform such as intercepting, deleting, modifying, forging or delaying packets.

1. *Packets and fields specification*: According to the communication protocol used by the WSN, exchanged packets must follows some rules in terms of packets size and fields value. Indeed, a malicious node can inject into the network malformed packets in order to lead receiving nodes to unstable state.

$$\begin{aligned} &Hold(wnet, s, sending, packet, not\_valid\_packet) \\ &\leftarrow \neg is\_ValidPacket(packet) \end{aligned} \tag{25}$$

2. *Forging a fake packet*: In this attack, the subject s forwards a packet o however the context *forwardingPacket* is not active. This means that the packet o is a packet forged by s that pretends forwarding a received packet.

$$\begin{aligned} Hold(wnet, s, a, o, forged\_packet) \leftarrow \\ empower(wnet, s, wForwarder) \wedge consider(wnet, a, sending) \\ \wedge use(wnet, o, packets) \\ \wedge \neg hold(s, a, o, forwardPacket) \end{aligned} \tag{26}$$

3. *Delaying a packet*: In this attack, the subject s forwards a received packet o after that the maximum forwarding time has expired.

$$\begin{aligned} Hold(wnet, s, a, o, delayed\_packet) \leftarrow \\ empower(wnet, s, wForwarder) \\ \wedge consider(wnet, a, sending) \wedge use(wnet, o, packets) \\ \wedge packet\_received(s, o, t) \wedge packet\_sent(s, o', t') \\ \wedge is\_forwadedVersion(o, o') \\ \wedge hold(wnet, s, a, o, ForwardPacket) \wedge (t + \delta) < t' \end{aligned} \tag{27}$$

where δ represents the maximal time a packet must be forwarded within since it is received.

4. *Deleting a packet*: In this attack, the subject s does not forward a received packet o within the defined time δ'.

$$\begin{aligned} Hold(wnet, s, a, o, deleting\_packet) \leftarrow \\ empower(wnet, s, wForwarder) \\ \wedge consider(wnet, a, sending) \wedge use(wnet, o, packets) \\ \wedge packet\_received(s, o, t) \wedge \neg packet\_sent(s, o, t') \\ \wedge is\_forwadedVersion(o, o') \\ \wedge hold(s, a, o, ForwardPacket) \wedge (t' < t + \delta') \end{aligned} \tag{28}$$

Thus, a packet is considered as deleted if it has not been forwarded within the time δ' (with $\delta < \delta'$). Between δ and δ' a packet not forwarded is considered as delayed.

5. *Modifying a packet*: In this attack, the subject s forwards a modified version of a received packet o that does not complain with the used communication protocol.

$$\begin{aligned} Hold(wnet, s, a, o, modified\_packet) \leftarrow \\ empower(wnet, s, forwarder) \wedge consider(wnet, a, sending) \\ \wedge use(wnet, o, packets) \\ \wedge packet\_received(s, o, t) \wedge packet\_sent(s, o', t') \\ \wedge is\_forwadedVersion(o, o') \\ \wedge hold(wnet, s, a, o, forwardPacket) \wedge is\_ValidPacket(o') \end{aligned} \tag{29}$$

7 Implementation and Evaluations

7.1 Implementation

To evaluate wIDS performances, we use WirelessHART NetSIM [15], a WirelessHART SCADA-based systems simulator. The simulated network is composed

of a network manager and 9 wireless sensors. We implement IDS-agents into 3 of them and launched randomly attacks.

7.2 WSN attacks implementation

We test the proposed wIDS towards the following well-known attacks on WSN [16]:

- Jamming attack: A malicious node disturbs transmissions of nearby nodes by emitting packets periodically or continuously.
- Denial of Service (DoS) attack: A malicious node overwhelms the targeted node by sending a great amount of packets that will not be able to receive legitimate packets.
- Sinkhole and blackhole attacks: A malicious node misleads routing algorithm by transmitting false information to the base station. Consequently, a part of the traffic will be redirected to the malicious node which can drop the packets partially (wormhole) or totally (blackhole).
- Hello Flood attack: A malicious node with a large transmission range can flood a large part of the network with this kind of packets. Nodes receiving these packets, will assume that the malicious node is in their transmission range and exhaust their battery life by trying to communicate with it.
- Selective forwarding attack: A malicious node chooses selectively to drop some packets and to not forward them to their final destination.
- Forced delay attack: A malicious node delays the forwarding of some packets which can have harmful consequences in WISN where processes are time sensitive.

Furthermore, we test it towards 2 attacks targeting wirelessHART networks:

- Sybil attack [17]: In this attack, a malicious insider node forges a fake *Disconnect* packet (Used by nodes to inform their neighbors that they are leaving the network), puts the targeted node as the packet source address and then authenticates it using the *Network Key* (Shared by all legitimate nodes). As results, receiving nodes erase the target node from their communication planning.
- Broadcast attacks [18]: In this attack, a malicious insider node uses its knowledge of the *Broadcast Session* (A Key used to encipher end-to-end packets broadcasted by the Network Manager to all nodes) for injecting into false commands. This attack is more harmful than the previous one as the attacker pretends to be the Network Manager and can change nodes configuration parameters.

7.3 Experimental results

As indicated in Table 1, wIDS detects all tested well-known attacks. Each of these attacks, is not compliant with one or several security rules.

Performed tests report 100% correct identification of malicious actions and less than 2% of false positives. Depending on which security rule is violated, false

positives is about 0% for sybil or broadcast attacks and about 5% for jamming, DoS or forced delay attacks. Indeed, first cited attacks are composed of actions that are clearly identified as malicious while the second cited attacks can be assimilated to transitory transmission perturbations such as interferences. This rate may be reduced by the use of a threshold.

Attacks	Detection Rules
Jamming	Rule (5), Rule (10), Rule (15)
Denial of Service (DoS)	Rule (10), Rule (15)
Sinkhole and blackhole	Rule (3)
Selective forwarding	Rule (17)
Hello Flood	Rule (3)
Forced delay	Rule (17)
Sybil	Rule (3)
Broadcast	Rule (17), Rule (22), Rule (23)

Table 1. Well-known Attacks Detection

7.4 Discussion

Previous results confirm the correctness of wIDS conception. They show that the normal behavior of wireless nodes can be modelized. As expected, the detection rate is 100% and depends highly of the accuracy of node normal behavior. By combining local and central detection, wIDS can be applied to networks of several sizes both in terms of nodes number and geographical area. Indeed, IDS-agents have the capabilities to detect basic malicious actions without any cooperation between them.

Also, by focusing on the detection of basic malicious actions, wIDS is able to detect known attacks as well as unknown ones and this without requiring any training phase.

8 Conclusion and Future Works

In this paper, we present wIDS an efficient intrusion detection system specially designed for enforcing wireless-based SCADA systems. It builds the normal behavior model of wireless nodes on the base of used protocol specification. Conducted tests confirm that wIDS is able to detect a large number of attacks with a low false-positive rate. These performances rely mainly on the quality of the nodes normal behavior model that depends on expert knowledges.

On the other hand, as tests were conducted in a simulated environments, some physical phenomenons were not considered. Indeed, WISN are expected to be deployed in industrial harsh environment characterized by wide temperature range, vibrations, reflections due to metallic structures, etc. Such an environment can impact communication reliability which can increase the false-positive rate.

References

1. Huitsing, P., Chandia, R., Papa, M., Shenoi, S.: Attack taxonomies for the modbus protocols. IJCIP **1** (2008) 37–44
2. Fovino, I.N., Carcano, A., Murel, T.D.L., Trombetta, A., Masera, M.: Modbus/dnp3 state-based intrusion detection system. In: 24th IEEE International Conference on Advanced Information Networking and Applications, AINA, Australia, IEEE Computer Society (2010) 729–736
3. Mitchell, R., Chen, I.: A survey of intrusion detection techniques for cyber-physical systems. ACM Comput. Surv. **46**(4) (2013) 55:1–55:29
4. Mitchell, R., Chen, I.: A survey of intrusion detection in wireless network applications. Computer Communications **42** (2014) 1–23
5. Silva, A.P.R.D., Martins, M.H.T., Rocha, B.P.S., Loureiro, A.A.F., Ruiz, L.B., Wong, H.C.: Decentralized intrusion detection in wireless sensor networks. In Boukerche, A., de Araujo, R.B., eds.: Q2SWinet'05 - Proceedings of the First ACM Workshop on Q2S and Security for Wireless and Mobile Networks, Montreal, Quebec, Canada, October 13, 2005, ACM (2005) 16–23
6. Roosta, T., Nilsson, D.K., Lindqvist, U., Valdes, A.: An intrusion detection system for wireless process control systems. In: IEEE 5th International Conference on Mobile Adhoc and Sensor Systems, MASS,USA, IEEE (2008) 866–872
7. Coppolino, L., D'Antonio, S., Romano, L., Spagnuolo, G.: An intrusion detection system for critical information infrastructures using wireless sensor network technologies. In: Critical Infrastructure (CRIS), 2010 5th International Conference on. (Sept 2010) 1–8
8. Shin, S., Kwon, T., Jo, G.Y., Park, Y., Rhy, H.: An experimental study of hierarchical intrusion detection for wireless industrial sensor networks. IEEE Transactions on Industrial Informatics **6**(4) (Nov 2010) 744–757
9. HART Communication Foundation: WirelessHART. `http://www.hartcomm.org`
10. Deji, C., Mark, N., Aloysius, M.: WirelessHART : Real-Time Mesh Network for Industrial Automation. Springer US (2010)
11. Kalam, A.A.E., Baida, R.E., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miege, A., Saurel, C., Trouessin, G.: Organization based access control. In: Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on. (June 2003) 120–131
12. Cuppens, F., Cuppens-Boulahia, N., Sans, T., Miège, A.: A formal approach to specify and deploy a network security policy. In: Formal Aspects in Security and Trust: Second IFIP TC1 WG1.7 Workshop on Formal Aspects in Security and Trust (FAST), an event of the 18th IFIP World Computer Congress, August 22-27, 2004, Toulouse, France. (2004) 203–218
13. Debar, H., Thomas, Y., Cuppens, F., Cuppens-Boulahia, N.: Enabling automated threat response through the use of a dynamic security policy. Journal in Computer Virology **3**(3) (2007) 195–210
14. Bayou, L., Cuppens-Boulahia, N., Espes, D., Cuppens, F.: Towards a CDS-based Intrusion Detection Deployment Scheme for Securing Industrial Wireless Sensor Networks. In: 11th International Conference on Availability, Reliability and Security, ARES 2016, Salzburg, Austria, August 31 - September 2, 2016, IEEE Computer Society (2016) 157–166
15. Bayou, L., Espes, D., Cuppens-Boulahia, N., Cuppens, F.: Wirelesshart netsim: A wirelesshart scada-based wireless sensor networks simulator. In Bécue, A.,

Cuppens-Boulahia, N., Cuppens, F., Katsikas, S.K., Lambrinoudakis, C., eds.: Security of Industrial Control Systems and Cyber Physical Systems - First Workshop, CyberICS 2015 and First Workshop, WOS-CPS 2015, Vienna, Austria, September 21-22, 2015, Revised Selected Papers. Volume 9588 of Lecture Notes in Computer Science., Springer (2015) 63–78
16. Karlof, C., Wagner, D.: Secure routing in wireless sensor networks: attacks and countermeasures. Ad Hoc Networks **1**(2-3) (2003) 293–315
17. Bayou, L., Espes, D., Cuppens-Boulahia, N., Cuppens, F.: Security issue of wirelesshart based SCADA systems. In Lambrinoudakis, C., Gabillon, A., eds.: Risks and Security of Internet and Systems - 10th International Conference, CRiSIS 2015, Mytilene, Lesbos Island, Greece, July 20-22, 2015, Revised Selected Papers. Volume 9572 of Lecture Notes in Computer Science., Springer (2015) 225–241
18. Bayou, L., Espes, D., Cuppens-Boulahia, N., Cuppens, F.: Security analysis of wirelesshart communication scheme. In Cuppens, F., Wang, L., Cuppens-Boulahia, N., Tawbi, N., García-Alfaro, J., eds.: Foundations and Practice of Security - 9th International Symposium, FPS 2016, Québec City, QC, Canada, October 24-25, 2016, Revised Selected Papers. Volume 10128 of Lecture Notes in Computer Science., Springer (2016) 223–238